САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

И МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Глушкова Екатерина Дмитриевна

Выпускная квалификационная работа бакалавра

Автоматизация формирования представления электронных схем цифровых устройств по их изображениям

Направление 010400

Прикладная математика и информатика

Научный руководитель,
кандидат технических наук,
доцент
Гришкин В. М.

Санкт-Петербург

2017

Содержание

Введение.

В современном мире невозможно представить себе отрасль производства, в которой та или иная задача решалась бы без применения вычислительной техники. И если изначально такая техника воспринимались лишь как второстепенное устройство – ведь большая часть работы лежала на человеке, то с развитием технологий на компьютеры перекладывается все больше и больше задач.

Изготовление современных электронных схем также не обходится без вычислительных технологий. В настоящее время системы автоматизированного проектирования (САПР) [1] активно используются при разработке схем, их тестировании и изготовлении. Каждую схему, смоделированную таким образом, можно редактировать и тестировать, а также использовать для создания более сложных устройств. Использование САПР на всех этапах производства позволяет повысить эффективность труда инженеров, в том числе:

- сокращает трудоемкость проектирования
- сокращает сроки проектирования
- сокращает себестоимость проектирования и изготовления
- повышает качество уровня результатов проектирования
- сокращает затраты на натуральное моделирование и испытание

Однако при модернизации ранее разработанных устройств и создании электронных архивов приходится сталкиваться с тем, что больше число схем было создано еще без использования компьютерных систем проектирования и существует лишь в виде чертежа на бумаге. Такое представление создает дополнительные трудности при разработке тестов устройств и переносе их на современную элементную базу.

На данный момент оцифровка таких схем осуществляется вручную. Это единственный гарантированный метод точного преобразования чертежа,

выполненного на бумаге, в формат, позволяющий работать с этими схемами используя системы автоматизированного проектирования. Однако эта процедура весьма длительная и трудоемкая.

В работе предлагается алгоритм, использующий методы компьютерного зрения для распознавания изображений схем цифровых устройств, позволяющий частично автоматизировать процесс перевода изображений таких схем в формат, использующийся в современной системе автоматизированного проектирования Quartus II [2,3].

Постановка задачи.

Электрическая схема – это документ, содержащий в виде условных изображений или обозначений составные части изделия, действующие при помощи электрической энергии, и их взаимосвязи.

Исходными данными (рис.1) для данной задачи являются изображения формата JPEG, полученные путем сканирования электронных схем (см. Приложение 1). Необходимо разработать алгоритм выделения объектов интереса на изображениях данного типа с последующим сохранением данных в документе формата BDF [4].

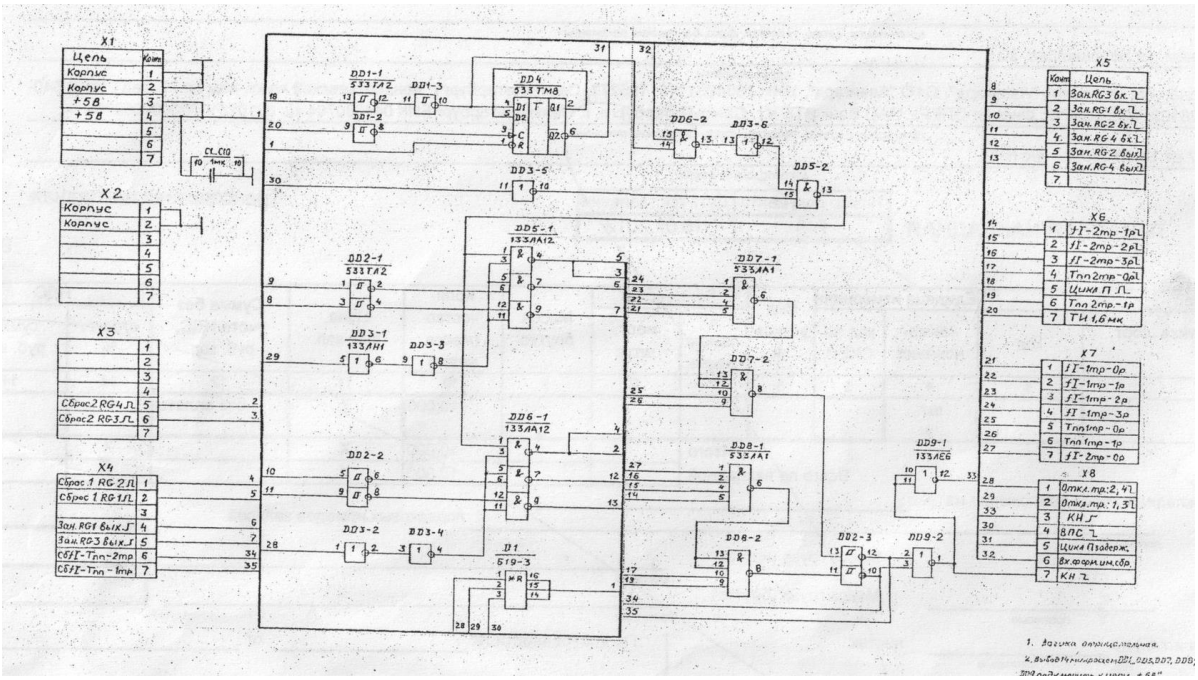


Рис.1 Пример исходного изображение электронной схемы

В качестве объектов интереса на данном изображении выступают такие части изображения как элементы схемы и линии взаимосвязи между ними.

Линии (рис.2) взаимосвязей на схеме, полученной в результате обработки, обозначаются как расположенные строго горизонтально или строго вертикально отрезки прямых, ведущие от одного компонента к другому, или к другим линиям взаимосвязей. В системе QuartusII обозначаются как connector.

В случае, если присутствует Т-образное пересечение линий взаимосвязей (рис.3), в месте пересечения присутствует точка связи, обозначаемая как junction:

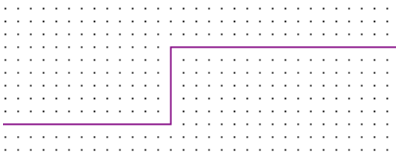


Рис.2 Connector

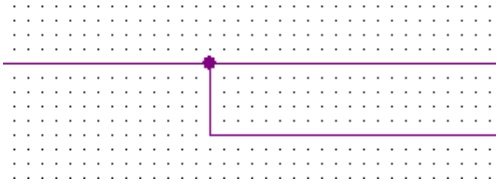


Рис.3 Junction

Элементы (рис.4) обозначены в виде связанных между собой прямоугольников произвольного вида (symbol). Прямоугольники могут располагаться как на расстоянии друг от друга, так и быть расположенными вплотную по вертикали. У каждого элемента имеются входные и выходные порты (port), к которым ведут линии соединения:

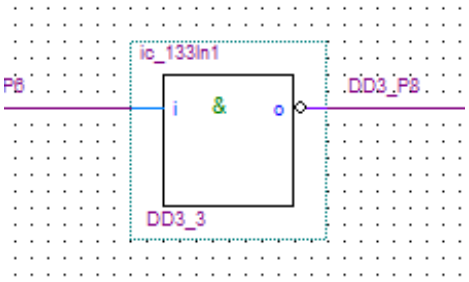


Рис.4 Symbol

Также на схеме присутствуют внешние входные (pin input) (рис.5) и выходные (pin output) цепи (рис.6). Они изображаются так же, как компоненты, но имеют всего один входной или выходной порт.

Рис.5 Pin Input

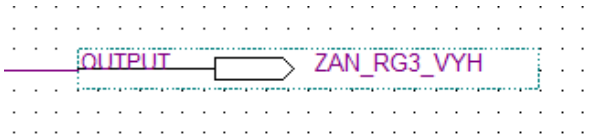


Рис.6 Pin Output

После проведения анализа начальных данных и результатов, которые должны быть получены, задача распознавания разбивается на 3 этапа:

1) предварительная обработка отсканированного изображения;

2) Поиск местоположения объектов интереса, а именно: линий связи с точками их пересечения, элементов, а также входных и выходных цепей вместе с их портами;
3) Сохранение полученных данных в формате BDF.

Глава 1. Предварительная обработка

На вход программе подается изображение формата JPEG. Изображение может быть как черно-белым, так и цветным. Схема должна целиком помещаться на этом изображении, быть оформленной согласна ГОСТ [5] и удобоваримо читаемой. Так как схема получена путем сканирования реальной бумажной схемы, допускаются такие искажения, как:

- Незначительные случайные повороты (не больше 30^0) и сдвиги;
- Следы сгибов (не повреждающие очертания схемы);
- Различные виды шумов.

Если на изображении присутствует рамка, диаграммы или таблицы – их лучше стереть, чтобы алгоритм не принял эти элементы за части схемы.

Так как на изображении много близко расположенных мелких объектов, любые его изменения, такие как предварительная фильтрация [6] с целью удаления шумов, могут испортить точность конечных результатов. Поэтому основной задачей предварительной обработки является удаление «отвлекающих объектов» с изображения и поворот его таким образом, чтобы линии связей располагались как можно ближе к строго горизонтальному и вертикальному направлению.

Для того, чтобы определить нужный угол поворота, будет создана копия исходного изображения, к которой уже применяются различные фильтры и методы нахождения произвольных кривых на изображении.

1.1 Получение монохромного изображения

В исходном виде изображение представлено матрицей пикселей, размерности $[M \times N]$. Элементами матрицы являются векторы размерности три, задающие цветовые характеристики каждого пикселя в виде значений цветового пространства RGB.

Для уменьшения вычислительных затрат, изображение (будь оно полноцветным или монохромным) заменяется аналогичным полутоновым изображением (рис.7) с помощью стандартной функции *cvtColor* [7] библиотеки openCV:

Каждый элемент матрицы заменяется на яркостное значение данного пикселя. Яркостная составляющая рассчитывается так:

$$Y = 0{,}299R + 0{,}587G + 0{,}144B,$$

где R, G и B, соответственно, красная, зеленая и синяя составляющая данного пикселя. Это изображение будет использовано в дальнейшем для детектирования точек интереса. Создаем его копию для дальнейшей обработки.

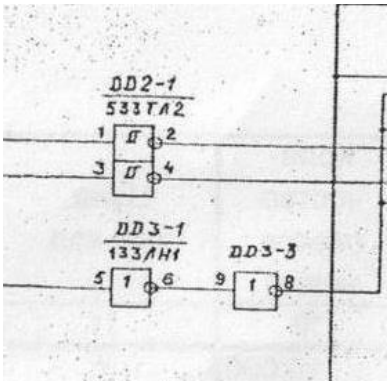


Рис.7
Результат работы *cvtColor*

1.2 Удаление шумов с изображения. Гауссово размытие.

Из-за того, что изображение получается путем сканирования, велика вероятность появления различных видов шумов. Как показывает практика, в основном это шум гауссовского типа. Этот шум практически не препятствует локализации элементов схемы, однако для легкости нахождения границ элементов изображения, необходимо избавиться от него и добавить размытие (рис.8). В условиях данной задачи со сглаживанием шумов лучше всего справляется алгоритм размытия по Гауссу, представленный в openCV как *GaussianBlur* [8]. Он выполняется путем свертки каждой точки входного массива с гауссовым ядром, а затем суммирования их всех для создания выходного массива.

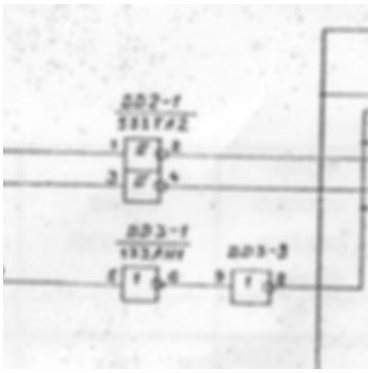


Рис.8
Результат работы *GaussianBlur*

Размытие по Гауссу – это фильтр размытия изображения, использующий нормальное распределение для вычисления преобразования, применяемого к каждому пикселю изображения.

При использовании данного фильтра каждый пиксель исходного изображения заменяется взвешенной суммой значений соседних пикселей. В ядре фильтра Гаусса веса распределяются по нормальному (гауссову) закону распределения:

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}},$$

Где параметр σ задает степень размытия

Данный фильтр хорошо работает для подавления краевых эффектов, так как веса пикселей задаются пропорционально близости к центру ядра.

1.3 Детектирование ребер. Алгоритм Кэнни.

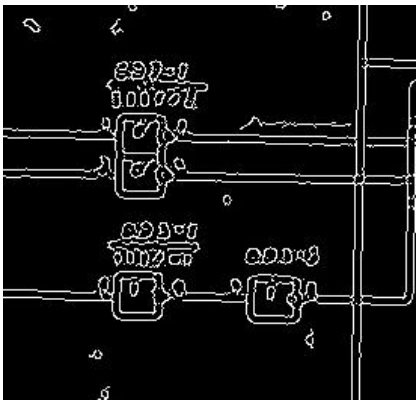


Рис.9
Результат работы *Canny*

Прежде чем применять методы нахождения произвольных кривых на изображении, необходимо выделить ряд точек «интереса», которые относятся к линиям. Для решения данной задачи применяется детектор границ Кэнни [9], реализованный в библиотеке openCV как *Canny* [10].

Границы – это такие кривые на изображении, вдоль которых наблюдается резкое изменение яркости. Детектор Кэнни находит точки, относящиеся к границам, основываясь на значениях градиентов каждого пикселя.

Для каждой точки изображения высчитывается градиент – двумерный вектор, компонентами которого являются производные яркости изображения по горизонтали и вертикали.

$$grad\ I(x,y) = (\frac{dI}{dx}, \frac{dI}{dy})$$

Получившийся в результате градиентный вектор будет ориентирован в направлении наибольшего увеличения яркости, а его длина будет соответствовать величине изменения яркости.

Для дальнейшего дифференцирования изображения используется так называемый оператор Собеля [11] – оператор, вычисляющий приближение градиента яркости изображения. Он основан на свертке изображения небольшими целочисленными фильтрами по вертикали и горизонтали. Результат применения данного оператора показывает, насколько резко изменяется яркость изображения в каждой точке: имеем значения первой производной в горизонтальном направлении (Gx) и вертикальном (Gy). Угол направления границы находится из этого градиента по формуле:

$$Q = \arctan(Gx/Gy)$$

Угол направления границы округляется до 0, 45, 90 или 135 градусов. Если величина градиента в данной точке достигает локального максимума в соответствующем направлении, то эта точка считается граничной (рис.9).

1.4 Выделение прямых на изображении. Алгоритм Хафа.

Алгоритм преобразования Хафа [12] используется для нахождения графических примитивов известной формы, например для поиска на изображении плоских кривых заданных параметрически.

Основная идея этого преобразования заключается в поиске прямых, которые проходят через достаточное количество точек интереса. Пускай существует семейство прямых на плоскости, заданных (рис.10) параметрическим уравнением:

$$F(R, \theta, x, y) = x\cos\theta + y\sin\theta - R$$

где $R -$ длина перпендикуляра опущенного на прямую из начала координат, θ - угол между перпендикуляром к прямой и осью 0X. Угол θ измеряется в пределах от 0 до 2π, а радиус R – ограничен размерами входного изображения. Параметры семейства прямых образуют фазовое пространство, каждая точка которого соответствует некоторой прямой.

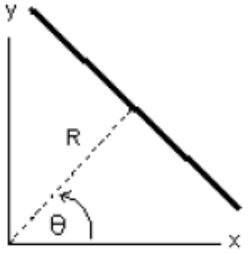


Рис.10 Вид задания прямой

Через каждую точку (x, y) изображения можно провести несколько прямых с разными R и θ. С другой стороны каждой точке фазового пространства (R, θ) будет соответствовать набор точек (x, y), образующих прямую на изображении. Из-за дискретности машинного представления изображения, все непрерывное фазовое пространство переводится в дискретное с помощью разбиения на сетку. При таком разбиении каждая ячейка соответствует набору прямых с близкими значениями параметров. В таком случае через каждую точку будет проходить конечное число прямых. Остается лишь каждой точке (R_0, θ_0) фазового пространства (R, θ) сопоставить счетчик, соответствующий количеству точек (x, y), лежащих на прямой

$$x\cos\theta_0 + y\sin\theta_0 = R_0$$

Данный метод реализован в библиотеке OpenCV в двух интерпретациях: как *HoughLines*, который осуществляет поиск прямых, и как *HoughLinesP*, который детектирует отрезки [13]. Из-за большого количества расположенных рядом объектов, а также из-за того, что надписи тоже воспринимаются алгоритмом Canny как точки интереса – предпочтительнее использовать именно вторую интерпретацию преобразования Хафа (Рис.11).

Рис.11 Результат работы *HoughLinesP*

После фильтрации отрезков прямых, полученных при выполнении алгоритма (удаления слишком коротких линий и тех, что находятся под большим наклоном), мы можем выделить средние значения двух самых популярных направлений линий на изображении (см. Приложение 2). Именно эти направления и будут являться реальными направлениями вертикали и горизонтали на схеме.

Далее осуществляется поворот (рис.12) на заданный угол с помощью функции *getRotationMatrix2D* [14] библиотеки openCV.

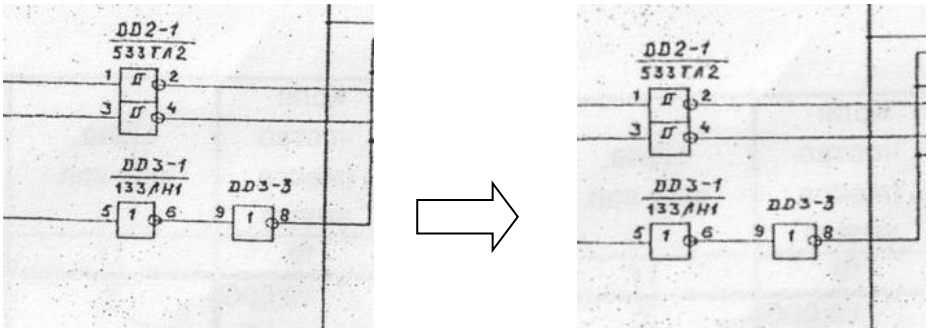


Рис.12 Результат работы getRotationMatrix2D

Таким образом мы получили изображение (см. Приложение 3), на котором в дальнейшем будет осуществляться поиск элементов схемы. Теперь линии в большинстве своем лежат строго горизонтально и вертикально, что позволит облегчить дальнейшую обработку и поиск элементов схемы.

Глава 2. Локализация объектов интереса.

В предыдущей главе уже использовались методы поиска линий и граничных точек, однако ни один из них не работает достаточно хорошо для данной задачи. При использовании Алгоритма Хафа с предварительным поиском границ методом Кэнни, находятся границы линий связи, а не сами эти линии, а подписи и условные обозначения выделяются как объекты сложной формы, что мешает распознаванию элементов схемы. В данной главе предложен алгоритм поиска прямых и элементов на изображении, специально адаптированный для данной задачи.

2.1 Выделение точек интереса

Первым шагом алгоритма станет выделение на изображении пикселей, являющихся частями линий связи. Линии, изображенные на схеме значительно темнее фона, имеют толщину в несколько пикселей, к тому же благодаря предварительной обработке, они близки к строго вертикальному и горизонтальному положению. Таким образом по цвету и местоположению в пространстве можно определить принадлежность данного пикселя к линии, используя окно поиска.

Так как наше изображение монохромное, его можно представить в виде матрицы размерности $[M \times N]$, каждый элемент которой характеризует яркостное значение заданного пикселя. Все изображение разбивается на небольшие, перекрывающие друг друга прямоугольные окна (рис.13), размером всего в несколько пикселей. Далее в каждом окне находятся те ряды и столбцы пикселей, которые значительно темнее средней яркости всего окна.

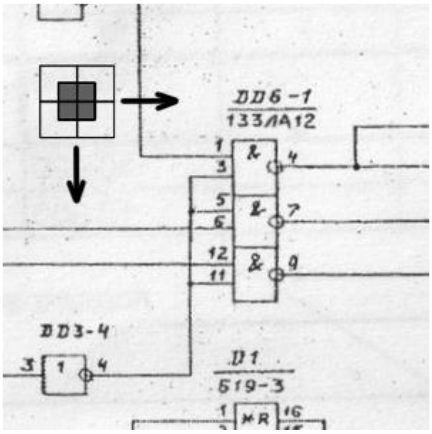


Рис.13
Окно поиска

Представим окно поиска как матрицу, размерности $[m \times n]$, в которой элемент $I_{i,j}$ – яркость соответствующего пикселя. Средняя яркость окна I_{sr} рассчитывается по формуле:

$$I_{sr} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{i,j}$$

Рассчитываем среднюю яркость каждого ряда и столбца из окна поиска:

$$I_j = \frac{1}{n} \sum_{j=1}^{n} I_{i,j}, \quad I_i = \frac{1}{m} \sum_{i=1}^{m} I_{i,j},$$

где I_j – средняя яркость j-того столбца, I_i – средняя яркость i-той строки.

Условие, что заданный столбец / строка является частью схемы:

$$I_j < kI_{sr}, \quad I_i < kI_{sr}, \quad 0 \leq k \leq 1$$

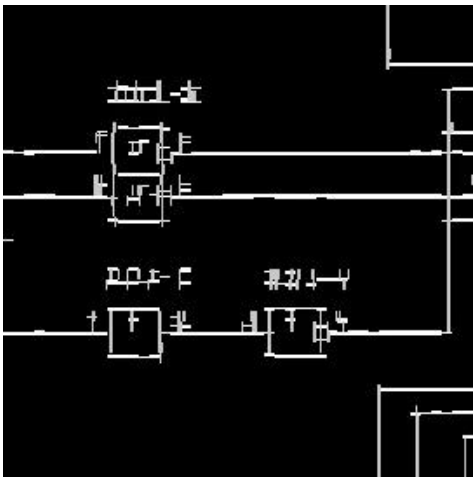


Рис.14 Найденные точки интереса

Таким образом пиксели, принадлежащие темным отрезкам, запоминаются как точки интереса. Важно сразу разделить пиксели, принадлежащие вертикальным и горизонтальным линиям на изображении. При этом некоторые пиксели могут принадлежать к обеим этим группам. В результате получаем две матрицы размерности исходного изображения, заполненные 0 и 1 (см. Приложение 4). Значение пикселя будет равно 1 если это точка интереса, 0 – если точка фона

2.2 Выделение связанных областей

В результате необходимо получить идеально ровные линии толщиной в один пиксель. Однако, так как линии на чертеже гораздо толще одного пикселя, то после работы этого алгоритма каждая линия будет представлена группой точек, расположенных на соседних строках и столбцах. На данном этапе среди точек интереса необходимо выделить группы связных областей пикселей, относящихся к одной и той же линии (см.Приложение 5).

Связанная область на изображении – такой набор точек, в котором две другие точки соединены друг с другом через последовательность соседей. Существует 2 вида определения связности:

1) 4-связность, когда соседями (рис.15) считаются точки, имеющие с данной точкой одну общую сторону.

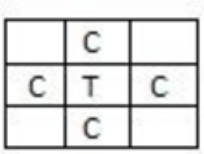


Рис.15 Маска 4-связности

2) 8-связность, когда соседями (рис.16) считаются точки, имеющие с данной точкой общую сторону или общий угол.

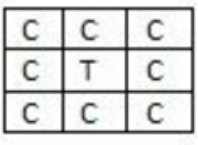


Рис.16 Маска 8-связности

В данной задаче необходимо использовать 8-связную модель поиска соседей, так как при использовании 4 связной некоторые группы пикселей, относящихся к одной линии, не будут восприниматься как соседи (рис.17).

Рис.17 Поиск соседей при использовании разного типа масок

Проходя по пикселям изображения, находим группы пикселей, принадлежащих одной линии. За счет того, что мы изначально разделили между собой точки интереса горизонтальных и вертикальных линий, их пересечения никак не повлияют на распознавание.

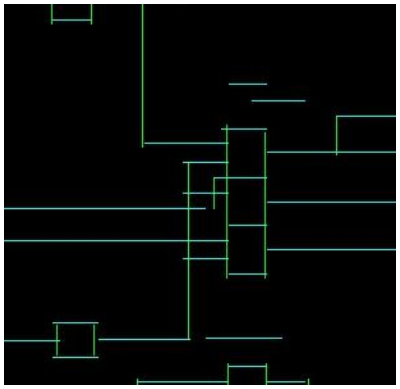


Рис.18
Распознанные линии

Рассмотрим вариант работы с группой пикселей, относящихся к одной горизонтальной линии. Среди каждой полученной группы пикселей находится самая высокая левая $(x1, y1)$ и самая низкая правая $(x2, y2)$ точка. Так как необходимо получить строго горизонтальную линию, значения по оси y усредняются, значения по x остаются прежними. Таким образом в результате получаем линию с координатами концов $\left(x1, \left\lfloor\frac{y1+y2}{2}\right\rfloor\right)$, и $\left(x2, \left\lfloor\frac{y1+y2}{2}\right\rfloor\right)$, округленными с точностью до целого.

2.3 Уточнение крайних точек линий.

Как видно на рис.18, найденные линии отвечают требованиям расположения строго вертикально и горизонтально, однако вследствие проведенных преобразований а также неточности передачи изображения при сканировании, на изображении остались некоторые типы искажений.

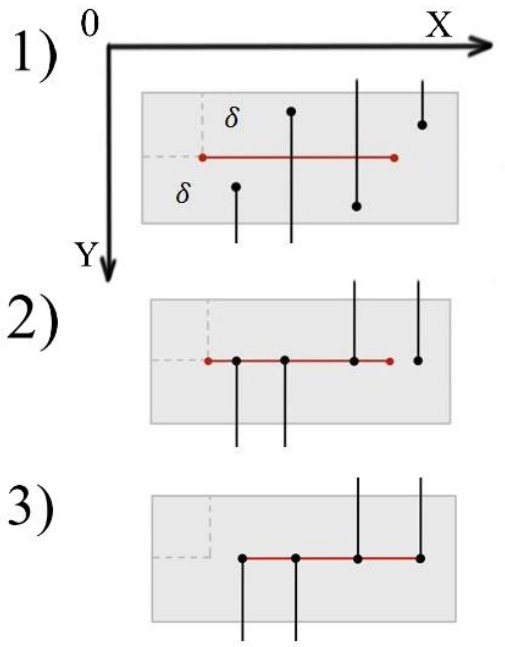


Рис.19
Уточнение линий

1) Крайние точки некоторых линий не доходят до пересечения с другими линиями связи или элементами схемы, либо напротив - заходят за границы. Линии соединений в общем случае должны быть непрерывны, поэтому следующим этапом станет уточнение их концов.

Для каждой горизонтальной линии $(x_1^0, y_1^0), (x_2^0, y_2^0)$, мы можем составить список вертикальных линий, которые возможно достроить до пересечения с ней. Программе задается δ - параметр, отвечающий за ширину окна поиска, которое строится вокруг заданной горизонтальной линии (рис.19).

Вертикальная линия $(x_1^1, y_1^1), (x_2^1, y_2^1)$, попавшая в данное окно любым своим концом, считается рекомендованной для уточнения. Пускай, для примера, в окно поиска попала начальная точка (x_1^1, y_1^1) вертикальной линии – это значит, что она удовлетворяет условиям

$$x_1^0 - \delta \leq x_1^1 \leq x_2^0 + \delta,$$
$$y_1^0 - \delta \leq y_1^1 \leq y_2^0 + \delta$$

Достраиваем вертикальные линии до пересечения. Координата точки, не попавшей в окно, остается неизменной, координата конца, попавшего в окно, заменяется координатой точки пересечения вертикальной и горизонтальной линии:

$$(x_1^1, y_1^1), (x_2^1, y_2^1) \quad \rightarrow \quad (x_1^1, y_1^0), (x_2^1, y_2^1)$$

Даже если после этого между линиями остается разрыв или перехлест, например, как на рисунке 17.2, они исчезают после применения аналогичного алгоритма, где за основу берется вертикальная линия, до которой достраиваются горизонтальные. Для наилучшего эффекта данный алгоритм необходимо повторить несколько раз как для горизонтальных, так и для вертикальных линий.

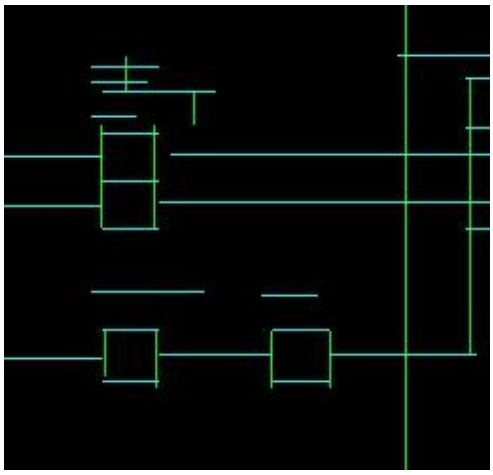


Рис.20 Линии, образованные подписями

2) В общем случае линии на изображении должны быть непрерывны и иметь пересечения на концах либо с другими линиями связи, либо с компонентами электрической схемы. Однако после использования предыдущего алгоритма все еще остаются линии, не имеющие пересечений. Например, больше количество «висящих в воздухе» прямых (рис.20), не имеющих пересечений ни на одном из концов. Велика вероятность того, что эти линии являются частью подписей или условных обозначений, принятых алгоритмом за часть схемы. Такие линии нам не нужны – смело их удаляем.

3) На изображении может присутствовать ряд горизонтальных и вертикальных линий, по какой-либо причине разделенных на несколько

частей. Если разрыв между ними не превышает некоторого порогового δ, заменяем две старые линии на одну новую, с усредненными координатами дальних концов изначальных линий.

Для горизонтальных:

$$\exists \ \ l_1\ (x_1^1, y_1^1), (x_2^1, y_2^1) \text{ и } l_2\ (x_1^2, y_1^2), (x_2^2, y_2^2),$$

$$\text{если } |x_2^1 - x_1^2| < \delta \ \ => l_3 \left(x_1^1, \left\lfloor \frac{y_1^1 + y_2^2}{2} \right\rfloor \right), \left(x_2^2, \left\lfloor \frac{y_1^1 + y_2^2}{2} \right\rfloor \right)$$

Для вертикальных:

$$\exists \ \ l_1\ (x_1^1, y_1^1), (x_2^1, y_2^1) \text{ и } l_2\ (x_1^2, y_1^2), (x_2^2, y_2^2),$$

$$\text{если } |y_2^1 - y_1^2| < \delta \ \ => l_3 \left(\left\lfloor \frac{x_1^1 + x_2^2}{2} \right\rfloor, y_1^1 \right), \left(\left\lfloor \frac{x_1^1 + x_2^2}{2} \right\rfloor, y_2^2 \right)$$

4) На изображении присутствуют короткие линии, лежащие достаточно близко к большим, параллельным им линиям (рис.21). Такие линии появляются из-за того что изначально толщина всех контуров достаточно велика по сравнению с пикселем, и слишком толстые линии могут восприниматься алгоритмом поиска как две разных. Алгоритм фильтрации таких линий похож на алгоритм достраивания линий до пересечения.

Программе задается h – параметр толщины окна поиска. Проходя по каждой линии $(x_1^0, y_1^0), (x_2^0, y_2^0)$, ищем, есть ли рядом с ней другие линии, полностью лежащие внутри окна поиска. Такая линия $(x_1^1, y_1^1), (x_2^1, y_2^1)$, должна удовлетворять условиям:

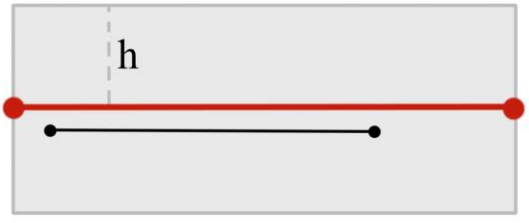


Рис.21 Близко расположенные короткие линии

$$x_1^0 - h \leq \ x_1^1 \ \leq x_2^0 + h,$$
$$y_1^0 - h \leq \ \ y_1^1 \ \leq y_2^0 + h,$$
$$x_1^0 - \ h \leq \ x_2^1 \ \leq x_2^0 + h,$$
$$y_1^0 - h \leq \ \ y_2^1 \ \leq y_2^0 + h,$$

Если условия выполняются, то удаляем эту короткую линию – скорее

всего это линия не является частью схемы.

5) После использования данных подходов велика вероятность того, что некоторые короткие линии свернутся в точку (начало и конец линии станут совпадать). Прежде чем продолжать распознавание, необходимо удалить все отрезки, не превышающие некоторого минимального значения

Каждый из этих алгоритмов решает определенные задачи и не дает полного избавления от неточностей. Однако при использовании их в комбинации друг с другом, повторяя несколько раз, варьируя коэффициенты и порядок их использования, можно добиться весьма хороших результатов. И хотя есть вероятность появления «мнимых» линий, сами линии не будут содержать ни разрывов, ни перехлестов (см. приложение 6).

2.4 Классификация объектов на изображении.

На данном этапе работы у нас имеются списки координат всех вертикальных и горизонтальных линий, обработанных таким образом, что они пересекаются с точностью до пикселя. В таких условиях задача нахождения прямоугольников [15], обозначающих компоненты схемы, точки пересечения линий связи а также входные и выходные порты компонент не составит труда.

1) Поиск прямоугольников

Найти прямоугольник на заданном изображении – значит найти 4 линии, каждая из которых стыкуется на концах с двумя другими линиями (координаты точек стыкуемых концов равны).

Проходя по рядам горизонтальных линий проверяем – может ли эта линия быть верхним ребром прямоугольника: находим, если есть, две стыкующееся с ней вертикальные линии, и если концы этих отрезков равноудалены от горизонтальной линии – запоминаем координаты предполагаемого левого верхнего и правого нижнего угла (рис.22). Затем выбираем из этих вертикальных линий левую: если существует горизонтальная линия, стыкующаяся с данной вертикальной в нижней точке, и правый конец этой горизонтальной линии совпадает с предполагаемым правым нижним концом прямоугольника, значит наше предположение было верно и координаты правого верхнего и левого нижнего концов можно сохранять как координаты элемента электросхемы.

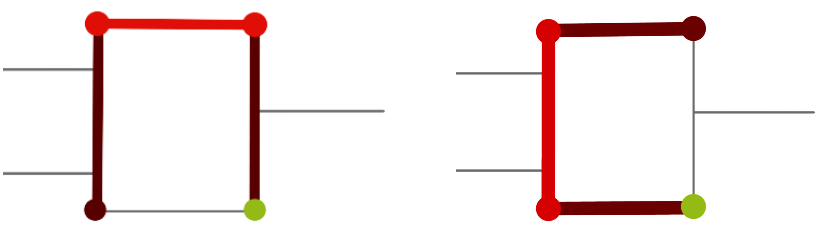


Рис.22 Поиск прямоугольников

Некоторые элементы располагаются один за другим по горизонтали, и при таком алгоритме поиска будут восприниматься как один. Для того, чтобы этого не произошло, необходимо найти все горизонтальные линии, лежащие внутри прямоугольника и расположенные ровно от левого его ребра до правого, и разделить прямоугольник на несколько новых прямоугольников, принимая за новые ребра эти самые горизонтальные линии.

Все прочие линии (вертикальные, короткие горизонтальные), находящиеся внутри полученных прямоугольников, необходимо удалить, так как, скорее всего, это части неотфильтрованных обозначений.

2) Поиск портов и точек пересечения линий

В схемах данного типа порты располагаются только с левой и с правой стороны элементов. Слева располагаются входные порты, справа – выходные. Проходя по списку горизонтальных линий, проверяем, лежит ли какой-либо ее конец на ребре прямоугольника, и если да – то на каком именно. В точке пересечения линии и ребра прямоугольника будет располагаться входной или выходной порт элемента. Их местоположение должно быть найдено для связи элементов и линий.

При Т-образном пересечении линий связи, образуется сложное пересечение. Местоположение точки такого пересечения необходимо локализовать для последующего построения схемы.

3) Классификация элементов.

После нахождения входных и выходных портов, относящихся к конкретным прямоугольникам, можно разделить все элементы на 3 категории:

- Внешние входные цепи (pin Input) – имеет только выходные порты.
- Внешние выходные цепи (pin Output) – имеет только входные порты
- Внутренние компоненты схемы (Symbol) – может иметь несколько входных и выходных портов

Таким образом результатом данного этапа стало получение достоверных сведений о местоположении элементов электронной схемы в пространстве а также о расположении линий связи между ними (см. приложение7).

Глава 3. Формирование выходных данных для САПР QuartusII.

В системе QuartusII каждая схема хранится в документе формата BDF. В этом документе формируется список объектов схемы с описанием их местоположения в пространстве. На этом этапе необходимо преобразовать данные, полученные при локализации компонентов схемы, в стандартный формат, использующийся данной САПР [16].

Лист для построения схемы может иметь неограниченно большой размер, поэтому преобразований координат не требуется и все компоненты можно размещать с теми же координатами, которые они имеют на печатной схеме.

1) Connector – линия связи.

В системе QuartusII линии связей изображаются ломанными, идущими от одного элемента схемы к другому. Ломанная состоит из отрезков (рис.23), каждый из которых задается в документе координатами его концов.

Линия, найденная на этапе детекирования:

$l = (x_1, y_1), (x_2, y_2)$

В документе будет иметь вид:

```
(connector
    (pt x1 y1)
    (pt x2 y2)
)
```

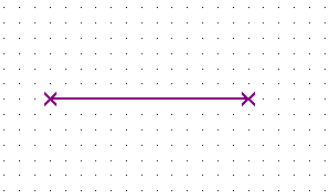


Рис.23 Connector

2) Junction – точка пересечения линий.

При построении схемы в системе Quartus вручную, точки пересечения линий связи ищутся системой автоматически, однако при преобразовании данных такие точки необходимо сохранять самостоятельно

Точка, найденная на этапе детектирования: $j = (x, y)$

Вид точки в документе: (junction (pt x y))

3) Pin Input, Pin Output – внешние входные и выходные цепи.

Входные и выходные цепи в системе Quartus представлены в виде вытянутых прямоугольников, содержащих информацию об имени и типе цепи, а также о местонахождении порта соединения (рис.24). Также внутри цепи расположен графический элемент, обозначающий входной / выходной порт. Так как все цепи однотипные, расположение элементов внутри прямоугольника задается в координатах относительны верхнего левого угла контура самого прямоугольника

Координаты прямоугольника: $(x_1, y_1), (x_2, y_2)$;

Координаты порта: (x_p, y_p);

Вид в документе:

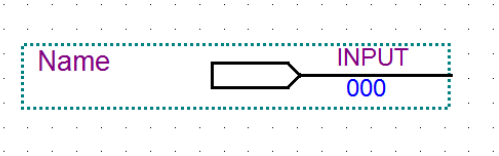


Рис.24 Pin Input

(pin
(input)
(rect x_1 y_1 x_2 y_2)
(text "INPUT" (rect 100 0 130 10)(font "Arial" (font_size 6)))
(text "Name" (rect 5 0 45 12)(font "Arial"))
(pt x_p y_p)
(drawing
(line (pt 60 14)(pt 87 14)(line_width 1))
(line (pt 60 6)(pt 87 6)(line_width 1))
(line (pt 90 10)(pt $(x_p - x_1)$ $(y_p - y_1)$)(line_width 1))
(line (pt 60 14)(pt 60 6)(line_width 1))
(line (pt 87 6)(pt 90 10)(line_width 1))
(line (pt 87 14)(pt 90 10)(line_width 1))
)
(text "000" (rect 106 10 126 19)(font "Arial" (font_size 6)))
)

Внешняя выходная цепь задается аналогично.

4) Symbol – внутренние элементы схемы.

Внутренние элементы схемы задаются в виде прямоугольников произвольного размера, содержащих информацию о названии и типе элемента, а также о количестве и местоположении входных и выходных

портов (рис.25). Внутри прямоугольника также расположены графические элементы, изображающие внутреннюю часть элемента, и линии связи с входными и выходными портами.

Координаты прямоугольника на схеме: $(x_1, y_1), (x_2, y_2)$

Координаты входного порта: (x_i, y_i)

Координаты выходного порта: (x_o, y_o)

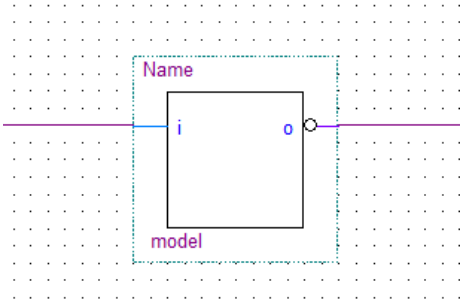


Рис.25 Pin Input

Вид в документе:

(symbol
(rect x_1 y_1 x_2 y_2)
(text "Name" (rect 5 0 50 12)(font "Arial"))
(text "model" (rect 8 80 40 92)(font "Arial"))
(port
(pt 0 $(y_i - y_1)$)
(input)
(text "i" (rect 18 $(y_i - y_1 - 4)$ 25 $(y_i - y_1 + 4)$)(font "Arial"))
(line (pt 0 $(y_i - y_1)$)(pt 16 $(y_i - y_1)$)(line_width 1))
)
(port
(pt $(x_2 - x_1)$ $(y_i - y_1)$)
(output)
(text "o"
(rect ($(x_2 - x_1 - 25)(y_i - y_1 - 4)(x_2 - x_1 - 18)$ $(y_i - y_1 + 4)$)
(font "Arial")
)
(line (pt $(x_2 - x_1 - 16)$ $(y_i - y_1)$)(pt $(x_2 - x_1)$ $(y_i - y_1)$)(line_width 1))
)
(drawing
(rectangle (rect 16 16 $(x_2 - x_1 - 16)$ $(y_2 - y_1 - 16)$) (line_width 1))
)
)

В результате сохранения такого документа в формате BDF-файла, получаем схему (см. Приложение 8), готовую для дальнейшей работы в QuartusII.

Реализация.

Реализация данного алгоритма выполнена на языке программирования Python. Этот язык выбран в основном благодаря существованию сторонних библиотек, предназначенных для манипуляций с матрицами и геометрическими объектами. Также существует несколько библиотек, отвечающих за работу с изображениями.

В данной работе активно используется библиотека openCV. В этой библиотеке содержится большое число полезных алгоритмов обработки изображений, компьютерного зрения и численных алгоритмов общего назначения.

Рис.26 Схема взаимодействия компонентов программы

Программа состоит из трех основных компонентов

1) Блок предварительной обработки.

Графическая информация, считываемая при помощи сканирования печатных схем, образует в памяти ЭВМ трехмерную матрицу $[M \times N \times 3]$, где M и N – длина и ширина изображения в пикселях. Внутри блока осуществляется нахождение угла наклона схемы и ее поворот до нужного положения. Выходными данными является трехмерная матрица повернутого изображения.

2) Блок распознавания элементов схемы.

На вход подается матрица результата предыдущего блока. В данном блоке выполняется последовательный поиск точек интереса, поиск линий и распознавание объектов на схеме. В процессе поиска точек интереса трехмерная матрица преобразуется в двумерную двоичную, где 1 означает

принадлежность к части схемы пикселя, соответствующего данной ячейке матрицы, а 0 – принадлежность фону. Далее получаем промежуточное описание, состоящее из списков координат вертикальных и горизонтальных линий. Данный список неоднократно переписывается в процессе уточнения линий. После выделения элементов из полученных линий формируются дополнительные списки, содержащие в себе координаты прямоугольников внутренних элементов, внешних входных и выходных контактов, а также точек пересечения линий и портов элементов схемы. Все списки объединяются в единый текстовый документ, на каждой строчке которого расположены координаты отдельного графического объекта. Все объекты расположены внутри своих групп, группы расположены в строгом порядке. Данный документ передается на следующий этап программы

3) Формирование выходных данных.

На данном этапе осуществляется замена значений координат элементов схемы на описательные структуры, предназначенные для распознавания системой QuartusII. Каждая текстовая строчка проверяется на соответствие некоторой группе объектов и заменяется ее «описанием» с добавлением координат данного объекта. Документ сохраняется в формате BDF-файла.

Вывод.

Алгоритм, предложенный в данной работе, решает задачу автоматизации процесса перевода печатных схем электронных устройств в формат, позволяющий работать с ними в системе QuartusII. Алгоритм справляется довольно неплохо: правильно локализует больше 90% элементов, расположенных на исходном изображении, классифицирует их и сохраняет полученные данные в формате BDF-файла.

Это позволяет проводить дальнейшую коррекцию распознанной схемы, используя средства САПР. Пользователю остается лишь внести некоторые ручные правки, проименовать контакты и заменить прямоугольники на типовые элементы, использующиеся при моделировании схемы.

Использование данного алгоритма позволяет значительно ускорить создание электронных схем и может рекомендоваться к использованию при создании электронных архивов.

Заключение.

В данной работе были рассмотрены вопросы распознавания изображений электронных схем цифровых устройств. Особое внимание уделено алгоритмам, позволяющим выделять на изображении структурные элементы схемы с определенной точностью. Найденные таким образом элементы служат основой для формирования файла в формате, используемом для САПР, что позволяет значительно ускорить процесс «оцифровки» схем, представленных в бумажном варианте и сократить трудозатраты при создании электронных архивов.

Список литературы

1. 1. А. Н. Корнилков Диагностические возможности САПР Quartus II фирмы Altera // Вестник Пермского университета, Вып 1(32), 2016, Стр. 22-28.

2. Valery M. Grishkin, Vladimir I. Melnik, Alexander N. Mikhailov. Methods of modeling of the test inputs for analysis the digital devices // International Conference on Computer Technologies in Physical and Engineering Applications, ICCTPEA 2014 – Proceedings Pages: 112-113, Year: 2014 DOI: 10.1109/ICCTPEA.2014.6893309

3. Гришкин В.М., Лопаткин Г.С., Михайлов А.Н., Овсянников Д.А. Интерфейсный метод построения моделей входных воздействий для тестирования электронных цифровых модулей // Вопросы радиоэлектроники. 2013, Том 1, №1 Стр. 80-89.

4. Kovshov A. Computer recognition of the electronic circuit drawing in hard copy //International Conference on Computer Technologies in Physical and Engineering Applications (ICCTPEA) 2014 pp. 75-76

5. ГОСТ 2.702-2011, Т52, «Единая система конструкторской документации (ЕСКД). Правила выполнения электрических схем»

6. Ковалевский В. А. Методы оптимальных решений в распознавании изображений // М Наука, 1976 – 328с

7. cvtColor. http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html

8. Smoothing Images. http://docs.opencv.org/2.4/doc/tutorials/imgproc/gausiga_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

9. Canny J. A computational approach to edge detection // IEEE Transactions on Pattern Analysis and machine intelligence, vol. PAMI-8, No. 6, P. 679 - 698

10. Canny openCV. http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html

11. Оператор Собеля. http://www.intuit.ru/studies/courses/10621/1105/lecture/17989?page=6

12. Duda R. O, Hart P. E. Use of the hough transformation to detect lines and curves in pictures // Commun. acm. 1972. Vol. 15, No 1. P. 11_15.

13. Hough transform. http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html

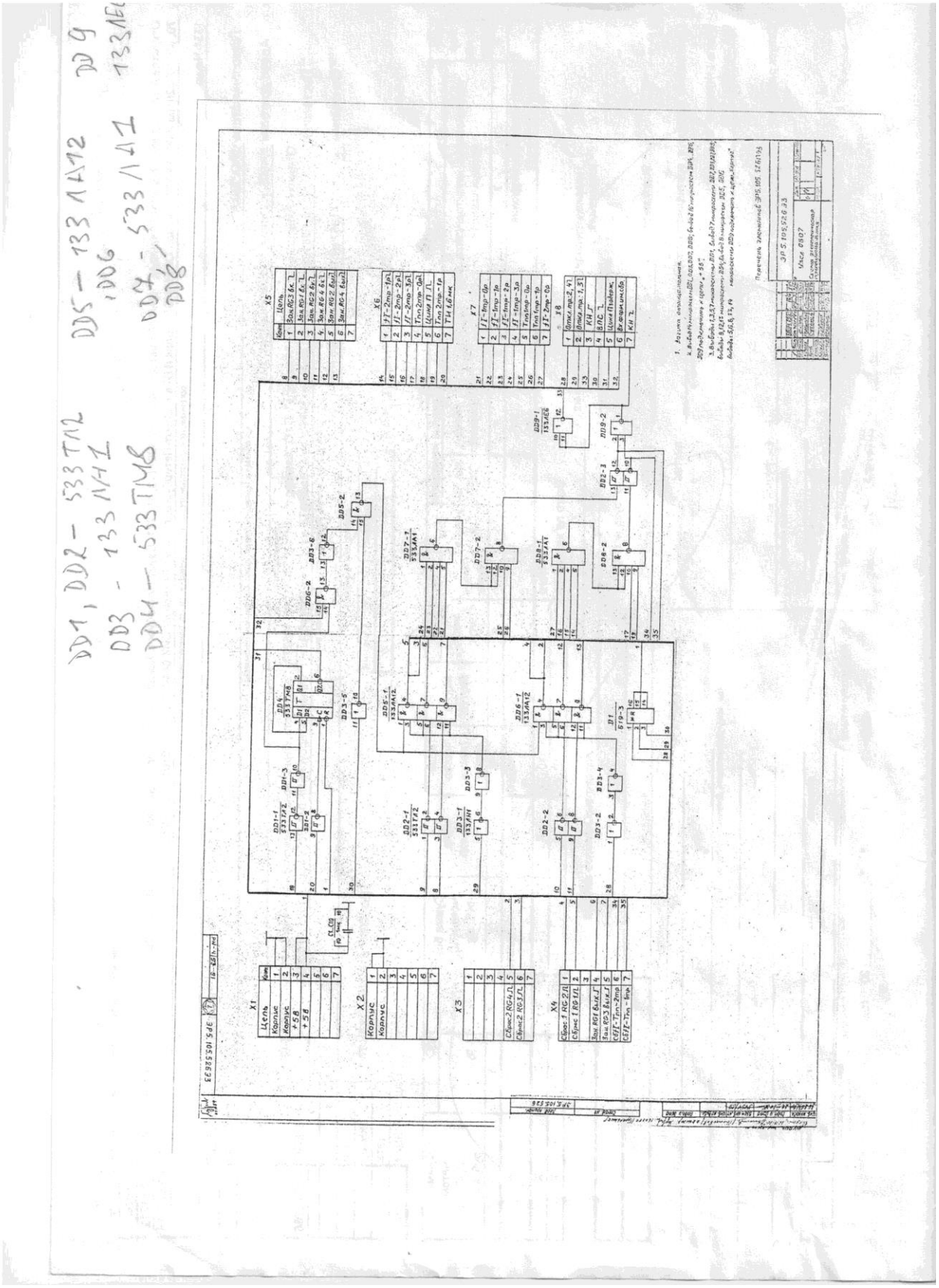
14. Geometric transformations of Images. http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html

15. Бутаков. Е. А. Обработка изображений на ЭВМ-Е //М. Радио и связь, 1987. С.134 – 159.

16. Ефремов Н. В. Введение в систему автоматизированного проектирования Quartus II: учебное пособие. М.: Изд-во ГОУ ВПО МГУЛ, 2011. 147 с

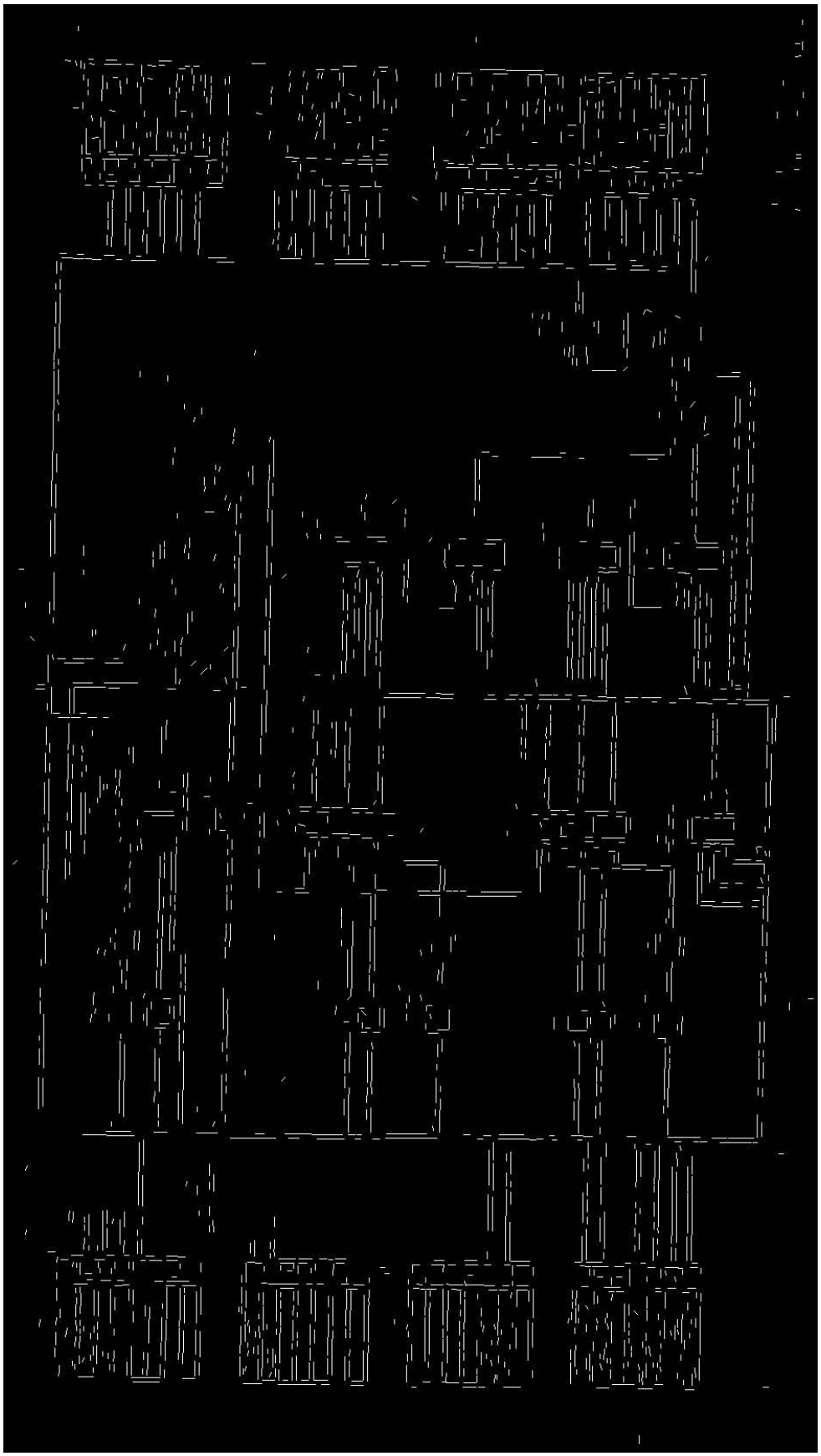
Приложение 1.

Скан чертежа электронной схемы



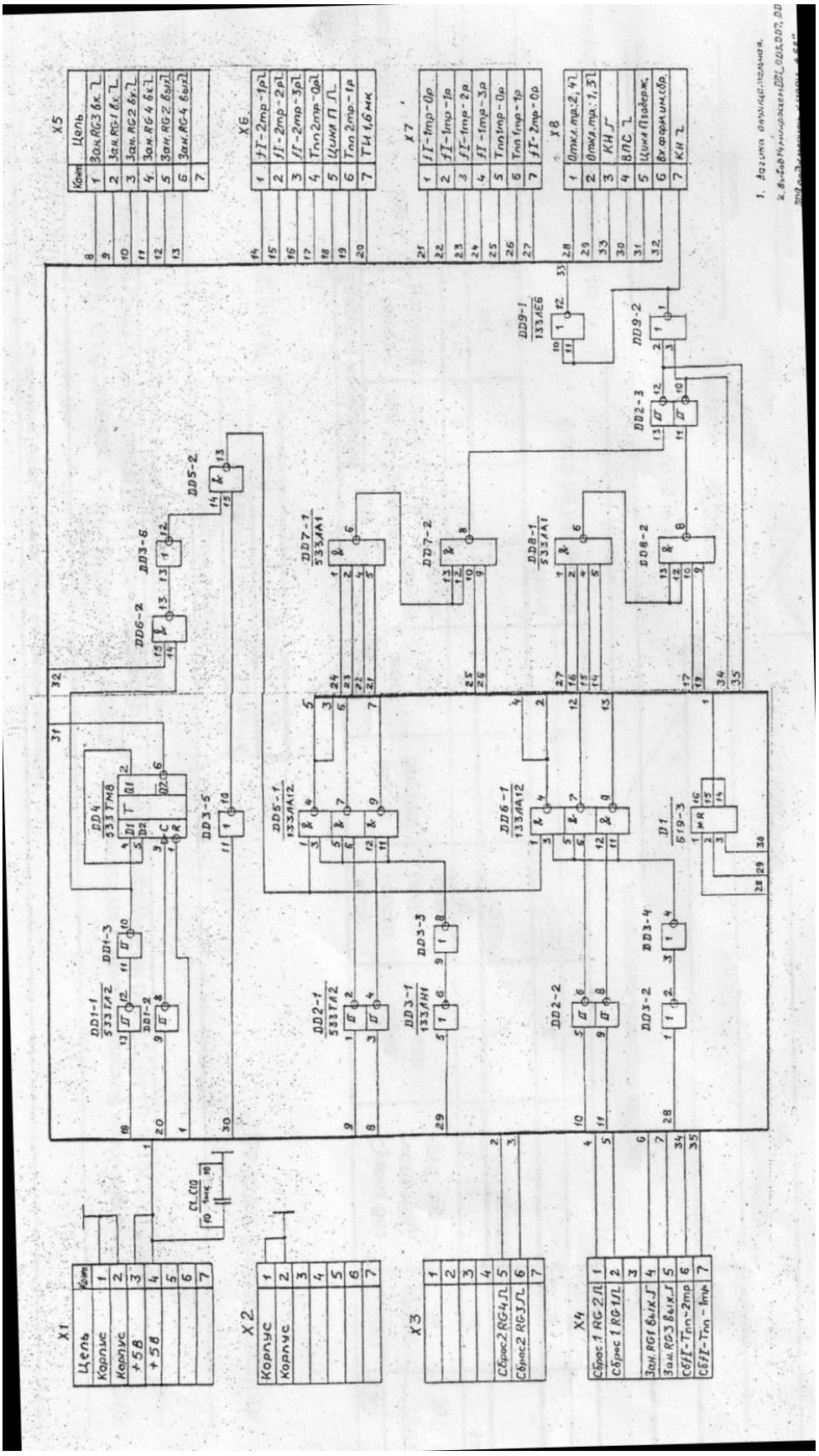
Приложение 2

Линии, найденные алгоритмом Хафа



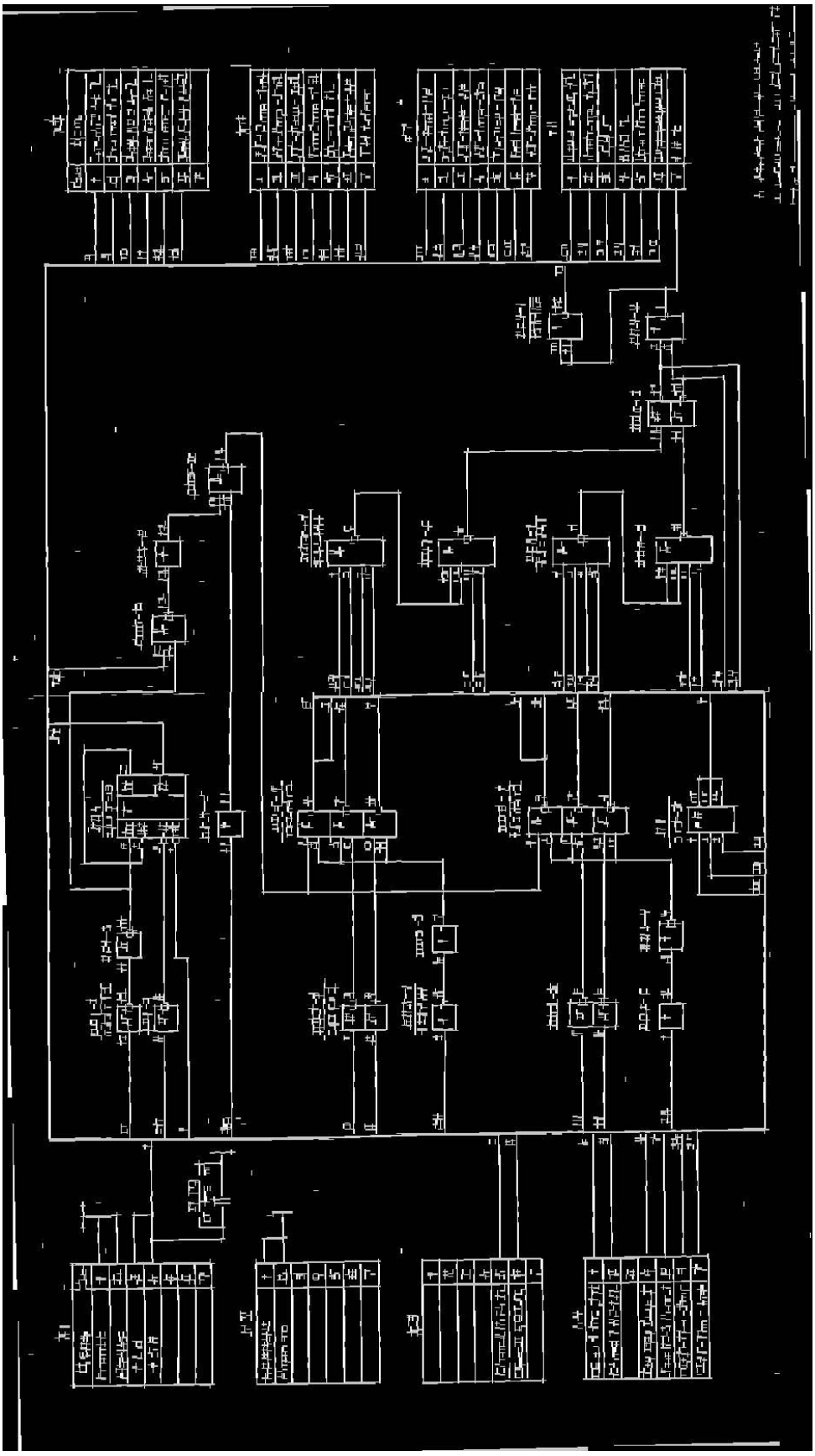
Приложение 3.

Изображение, прошедшее предварительную обработку.



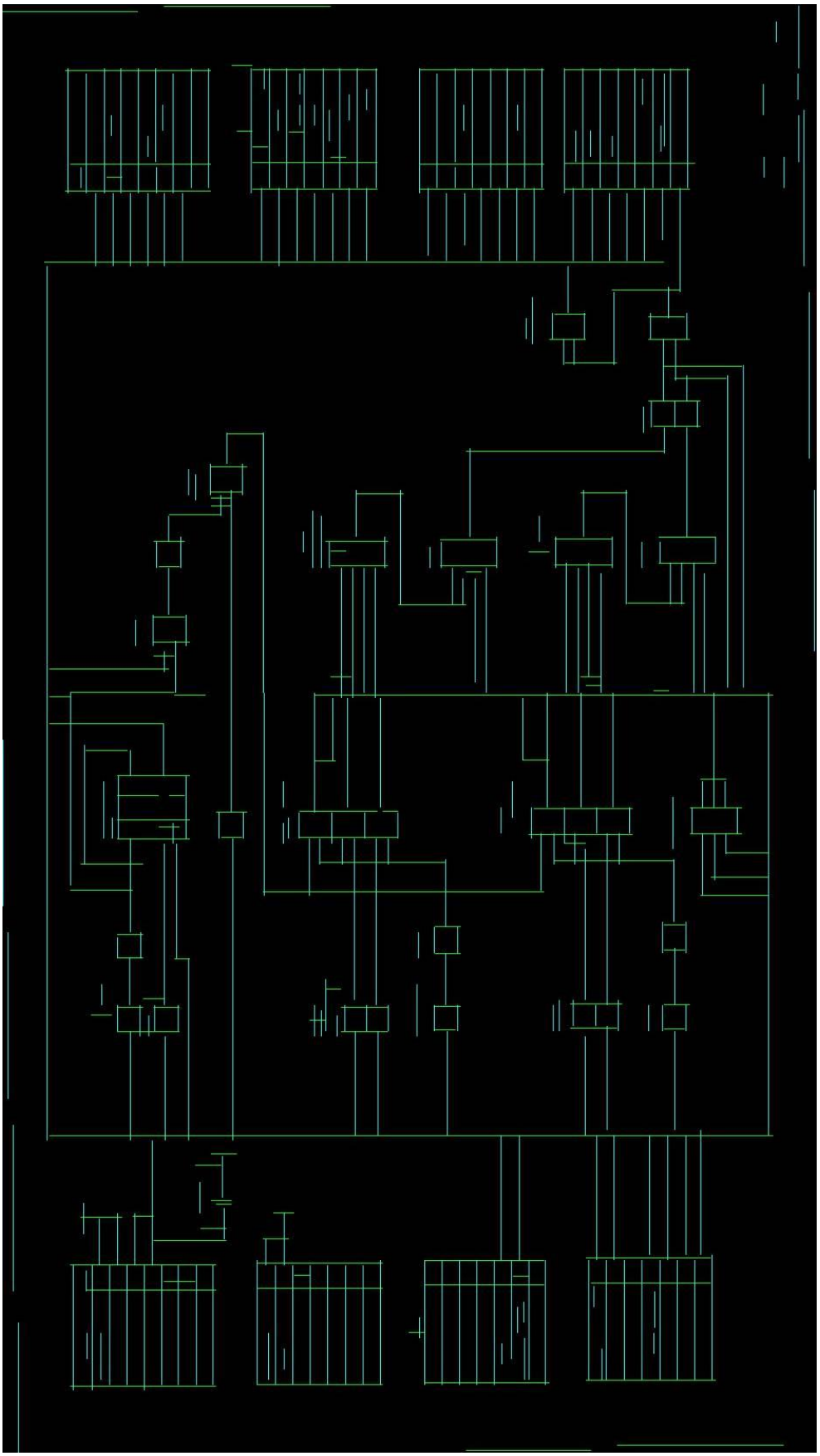
Приложение 4

Выделение точек интереса



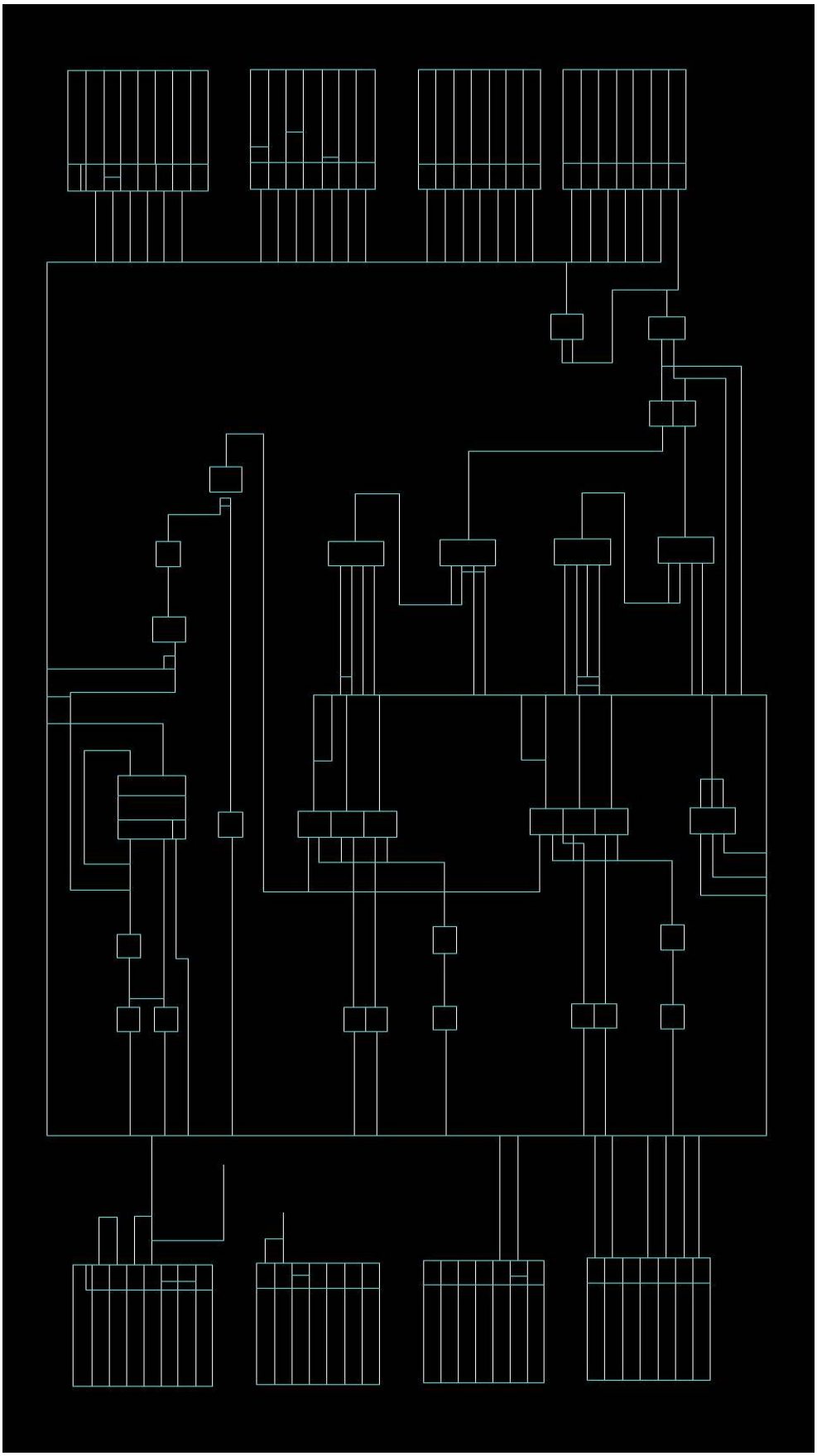
Приложение 5.

Выделение группы связных точек.



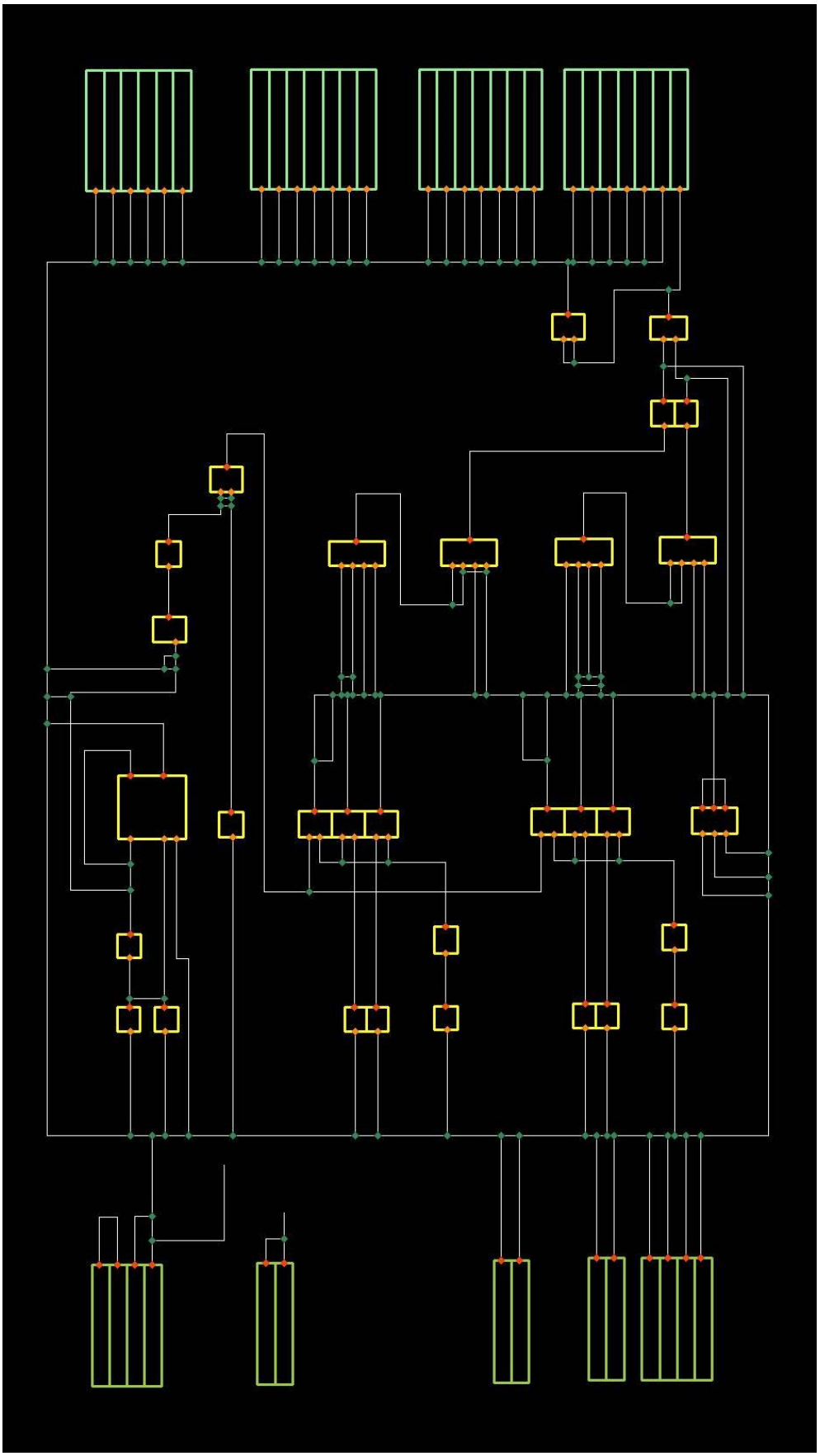
Приложение 6.

Результат применения методов исправления линий



Приложение 7.

Вид электронной схемы с распознанными объектами



Приложение 8.

Вид электронной схемы в системе QuartusII